Python – ESP32 (Aula 3)

Instrumentação Eletrotecnia Aplicada

IEA 2021-2022

A Borges

Conceitos gerais

- Python
- Python instalação

- IDE : Integrated Development Environment

(Ambiente de Desenvolvimento Integrado)

Python

- Linguagem generalista, possível de utilizar num vasto conjunto de disciplinas: biologia, química, finanças, analise numérica, robótica, etc;
- www.python.org

- Linguagens de alto nível (C, C++, Visual Basic, etc)
- Linguagens interpretadas (Python, Matlab, etc)
- Linguagens gráficas (LabVIEW)

Thonny - Instalação

1. Instalação Python (Windows)

https://www.python.org/downloads/release/python-3102/

2. Instalação do IDE Pythom-ESP32

https://thonny.org

MicroPython (Thonny) - Instalação

3. Instalar o **Drive do ESP32**

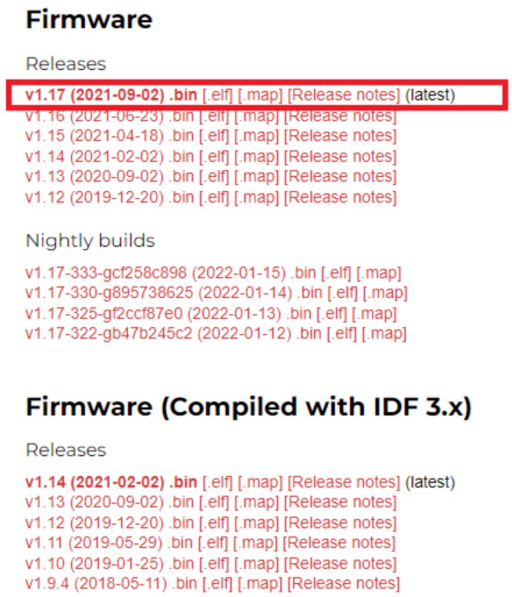
https://www.silabs.com/developers/usb-to-uart-bridge-vcp-drivers

CP210x Windows Drivers

Confirmar no **Gestor de Dispositivos** se está corretamente instalado e associado a uma porta série (**COM3**... Por exemplo)

4. Dowload **Firmware (ESP32)**

https://micropython.org/download/esp32/



5. Instalação plug-ins

Tools -> Manage plug-ins -> esptool

https://pypi.org/project/esptool/

na linha de comandos do Windows, instalar o Esptools

pip install esptool

6. Selecionar **MicroPython(ESP32)**

Tools -> Interpreter -> MicroPython(ESP32)

Selecionar a porta e eventualmente realizar o upload do firmware

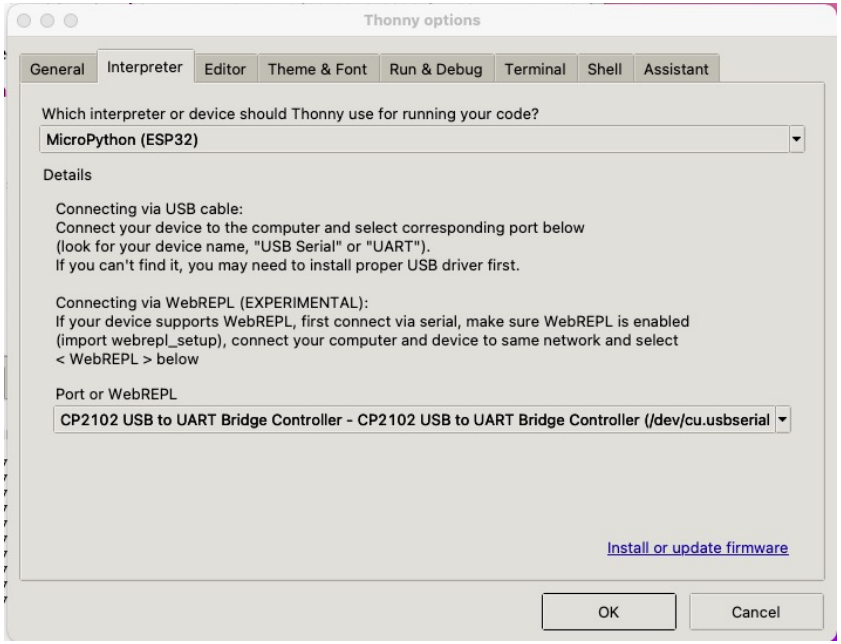
Python – ESP32 (Aula 3)

MicroPython

IEA 2021-2022

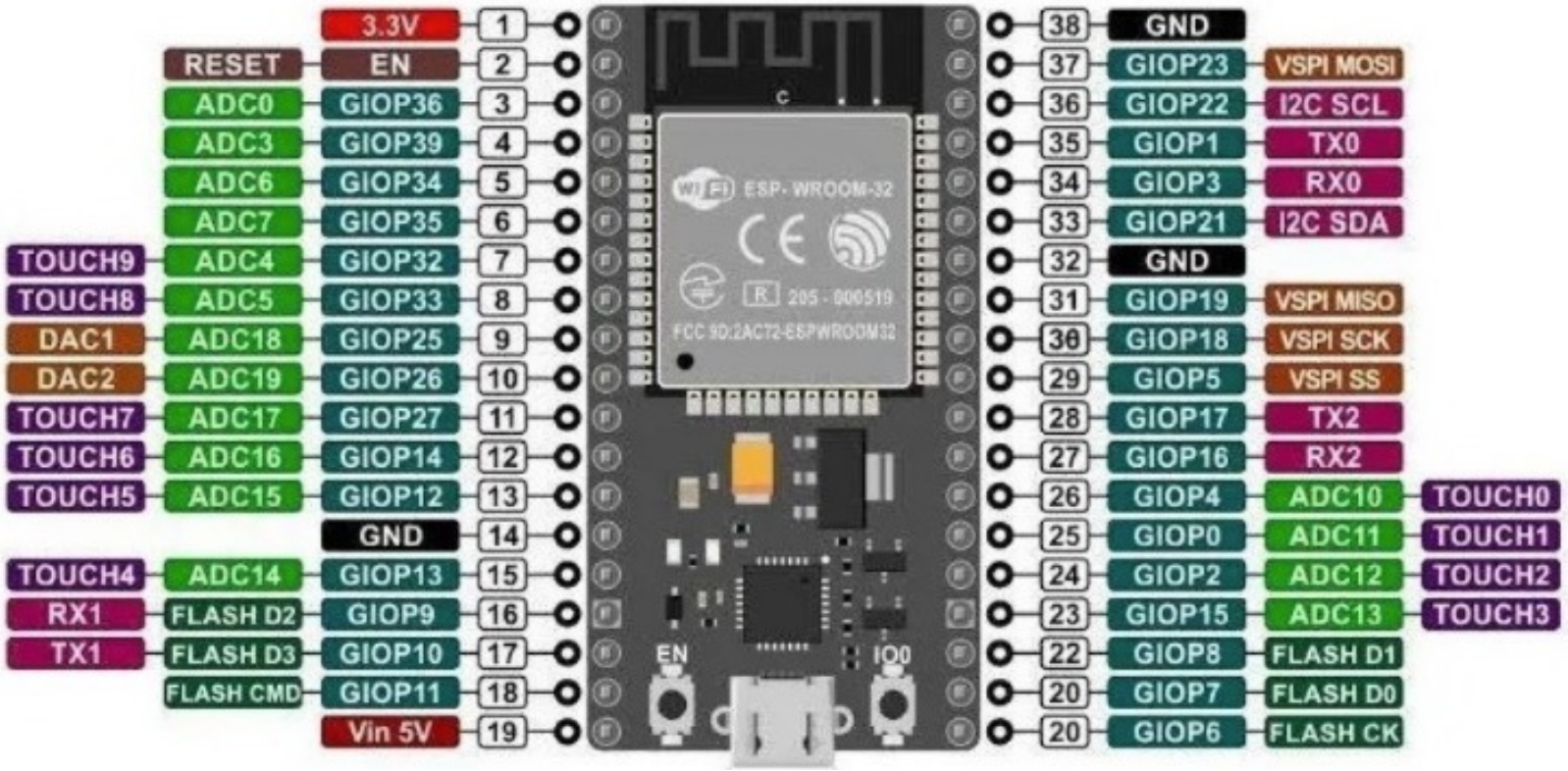
A Borges

MicroPyhon (ESP32):



Tools -> Options... -> Interpreter -> Seleccionar MicroPython (ESP32)

MicroPyhon (ESP32) : Diagrama



Resumo Instruções

print ('Hello World')
print ('\n')

A = **input** ('Qual o seu animal de estimação?')

Funções de conversão: **int(), float(), str(), bool()**

Operações matemática: **+ - * / //**(div inteira) **%** (resto) ****** (exp)
Operadores lógicos: **not and or**
Instruções de comparação: **== != > < >= <=**

```
if <cond 1>:
   #
elif <cond 2>:
   ...
else:
   ...
```

```
try:
   ....
except:
   ....
else:
   ....
```

```
while <cond>:
   ....
    break
```

```
def <nomefunção> (param in):
    # função
    return <par1, par 2, etc>
```

```
x1, x2, etc = nomefunção (par in)
```

```
for aux in range(1,10,2):
    print (aux)
```

```
from time import sleep
sleep(1)
```

```
from datetime import datetime
now = datetime.now()
current_time = now.strftime("%Y-%M-%d  %H:%M:%S")
```

```
# Abertura ficheiro
f = open ("Demo2.txt","w") # outros acessos: a, x
f.write ('Escreve uma linha em ficheiro de texto... \n')
f.write ('escreve outra linha em ficheiro de texto... \n')
f.close

# Visualiza o que escreveu em ficheiro de texto
f = open ("Demo2.txt","r")
xx = f.read()
print ( xx )
f.close
```

Leitura data e hora (MicroPython)

import utime as time

Read date and time

aux = **time.gmtime()**

print (aux)

format date and time

name = '{:04d}-{:02d}-{:02d} {:02d}:{:02d}:{:02d} '.format(aux[0],aux[1],aux[2],aux[3],aux[4],aux[5],)

print (name)

[Resultado]

>>> print (aux)

(2022, 3, 12, 20, 20, 54, 5, 71)

Exemplo 1

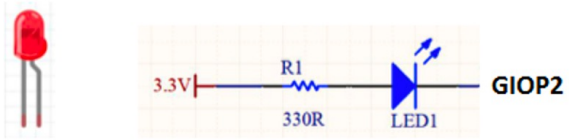
```
# Leitura da Data-Hora MicroPython
# 12 Março 2022

import utime as time

now = time.gmtime()
ff = '{:04d}-{:02d}-{:02d}  {:02d}:{:02d}:{:02d}'.format(now[0],now[1],now[2],now[3],now[4],now[5])
print ( ff )
```

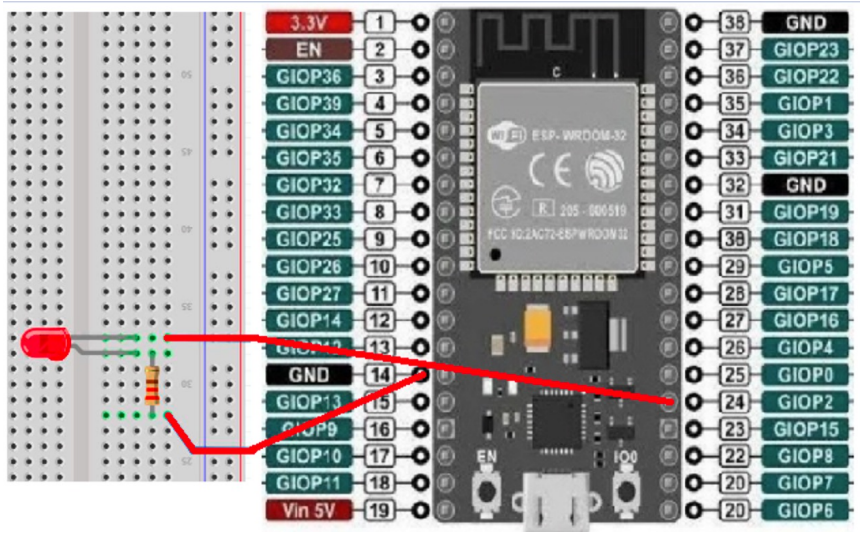
MicroPyhon: Pisca Pisca

Exemplo 2

```
# Led pisca/pisca
# 19 Março 2022

from time import sleep
from machine import Pin


# configuração pin de saída
led = Pin ( 2,Pin.OUT )
led.value ( True )


xx = input ( 'Digite uma tecla para iniciar :' )
aux = 0
while True:
    led.value ( not led.value() )
    sleep ( 0.5 )
    aux = aux + 1
    print ( aux, ' -> ', bool(led.value()) )
```

```
>>> %Run -c $EDITOR_CONTENT
  Digite uma tecla para iniciar :
  1  ->  False
  2  ->  True
  3  ->  False
  4  ->  True
  5  ->  False
  6  ->  True
  7  ->  False
  8  ->  True
  9  ->  False
  10  ->  True
  11  ->  False
  12  ->  True
  13  ->  False
  14  ->  True
  15  ->  False
```



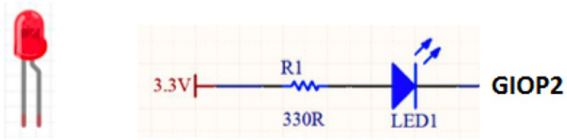


Notas:

Pisca de 0,5 segundos – ciclo infinito

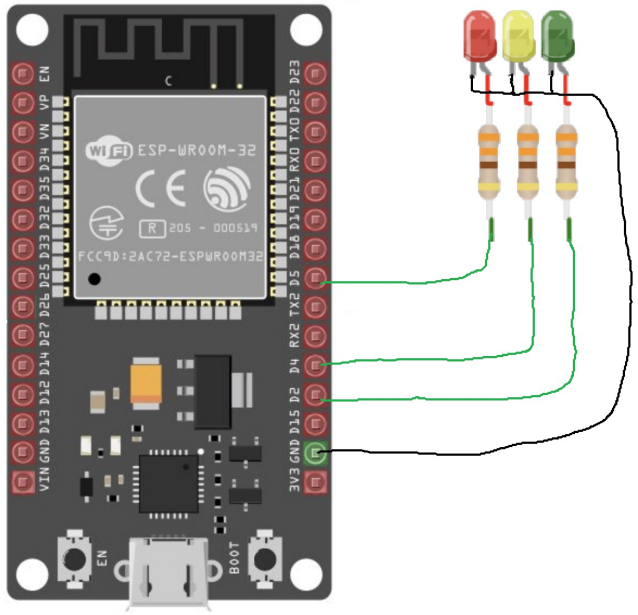
MicroPyhon: Semáforos (Exercício 3)

Funcionamento do Semáforo:



Implementar um semáforo:

- Verde (GPIO 2): 10 segundos
- Amarelo (GPIO 4): 2 segundos
- Vermelho (GPIO 5): 3 segundos



Utilizar um contador (aux) e uma pausa de 0,5 segundo (sleep)

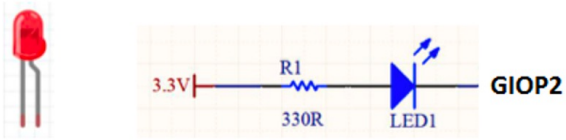
Apresentar na janela de comando o estado dos semáforos:

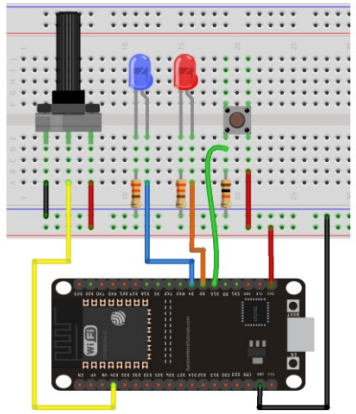
```
MicroPython v1.18 on 2022-01-17; ESP32 module with ESP32
Type "help()" for more information.
>>> %Run -c $EDITOR_CONTENT

  Digite uma tecla para iniciar...
  Estado : 01 - S Verde : True - S Am : False - S Vermelho : False
  Estado : 02 - S Verde : True - S Am : False - S Vermelho : False
  Estado : 03 - S Verde : True - S Am : False - S Vermelho : False
  Estado : 04 - S Verde : True - S Am : False - S Vermelho : False
  Estado : 05 - S Verde : True - S Am : False - S Vermelho : False
  Estado : 06 - S Verde : True - S Am : False - S Vermelho : False
  Estado : 07 - S Verde : True - S Am : False - S Vermelho : False
  Estado : 08 - S Verde : True - S Am : False - S Vermelho : False
  Estado : 09 - S Verde : True - S Am : False - S Vermelho : False
  Estado : 10 - S Verde : True - S Am : False - S Vermelho : False
  Estado : 11 - S Verde : True - S Am : False - S Vermelho : False
  Estado : 12 - S Verde : True - S Am : False - S Vermelho : False
  Estado : 13 - S Verde : True - S Am : False - S Vermelho : False
  Estado : 14 - S Verde : True - S Am : False - S Vermelho : False
  Estado : 15 - S Verde : True - S Am : False - S Vermelho : False
  Estado : 16 - S Verde : True - S Am : False - S Vermelho : False
  Estado : 17 - S Verde : True - S Am : False - S Vermelho : False
  Estado : 18 - S Verde : True - S Am : False - S Vermelho : False
  Estado : 19 - S Verde : True - S Am : False - S Vermelho : False
  Estado : 20 - S Verde : True - S Am : False - S Vermelho : False
  Estado : 21 - S Verde : True - S Am : False - S Vermelho : False
  Estado : 22 - S Verde : False - S Am : True - S Vermelho : False
  Estado : 23 - S Verde : False - S Am : True - S Vermelho : False
  Estado : 24 - S Verde : False - S Am : True - S Vermelho : False
  Estado : 25 - S Verde : False - S Am : True - S Vermelho : False
  Estado : 26 - S Verde : False - S Am : True - S Vermelho : False
```

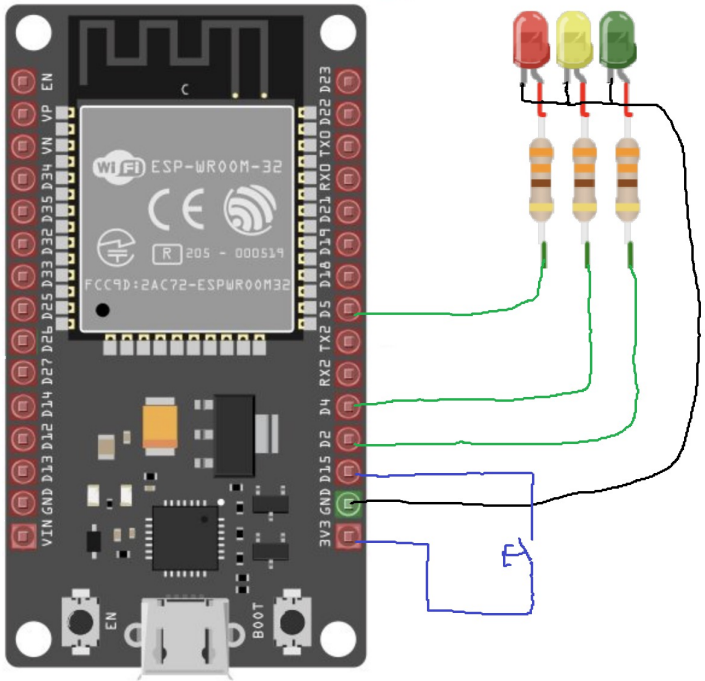
MicroPyhon: Leitura Entradas Digitais

Exemplo 4

```
Leitura entrada Digital
19 Março 2022

rom machine import Pin
rom time import sleep

Config Pin - PULL_DOWN (desligado)
utton = Pin (15, Pin.IN, Pin.PULL_DOWN )

hile True:
   sleep (0.5)
   if button.value():
       xx = 'Ligado'
   else:
       xx = 'Desligado'
   print ( 'Estado entrada 15 (GPIO15) : {:1d} - {}'.format(button.value(),xx) )
```








```
>>> %Run -c $EDITOR_CONTENT

   Estado entrada 15 (GPIO15) : 1 - Ligado
   Estado entrada 15 (GPIO15) : 1 - Ligado
   Estado entrada 15 (GPIO15) : 1 - Ligado
   Estado entrada 15 (GPIO15) : 1 - Ligado
   Estado entrada 15 (GPIO15) : 1 - Ligado
   Estado entrada 15 (GPIO15) : 1 - Ligado
```

MicroPyhon: Semáforos (Exercício 5)

Funcionamento do Semáforo:

Implementar um semáforo:

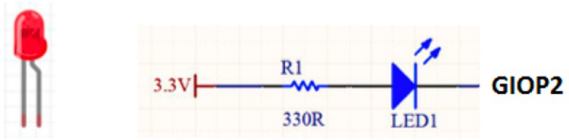
- Verde (GPIO 2): 10 segundos
- Amarelo (GPIO 4): 2 segundos
- Vermelho (GPIO 5): 3 segundos

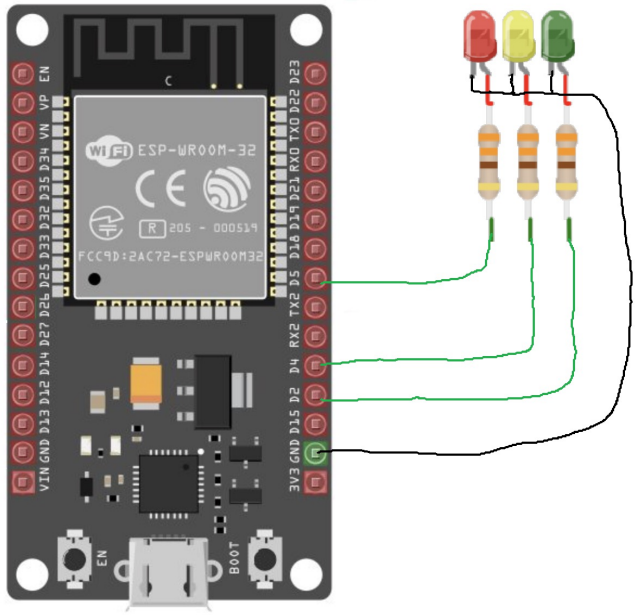
Utilizar um contador (aux) e uma pausa de 0,5 segundo (sleep)

Apresentar na janela de comando o estado dos semáforos:

Alterar o exercício 3, para funcionar apenas enquanto a entrada GPIO 15 – estiver ligada.

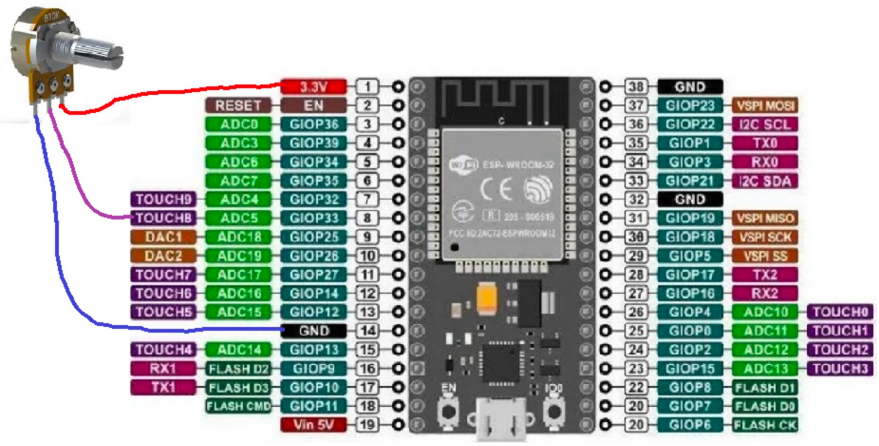
O ciclo só deve terminar, quando o contador (Estado)>30 e a entrada GPIO 15 estiver desligada. Utilizar uma variável booleana auxiliar (ciclo)






```
>>> %Run -c $EDITOR_CONTENT

  Digite uma tecla para iniciar...
  Estado : 01 - Button 1 (Ligado) - S Verde : True - S Am : False - S Vermelho : False
  Estado : 02 - Button 1 (Ligado) - S Verde : True - S Am : False - S Vermelho : False
  Estado : 03 - Button 1 (Ligado) - S Verde : True - S Am : False - S Vermelho : False
  Estado : 04 - Button 1 (Ligado) - S Verde : True - S Am : False - S Vermelho : False
  Estado : 05 - Button 1 (Ligado) - S Verde : True - S Am : False - S Vermelho : False
  Estado : 06 - Button 1 (Ligado) - S Verde : True - S Am : False - S Vermelho : False
  Estado : 07 - Button 1 (Ligado) - S Verde : True - S Am : False - S Vermelho : False
  Estado : 08 - Button 1 (Ligado) - S Verde : True - S Am : False - S Vermelho : False
  Estado : 09 - Button 1 (Ligado) - S Verde : True - S Am : False - S Vermelho : False
  Estado : 10 - Button 1 (Ligado) - S Verde : True - S Am : False - S Vermelho : False
  Estado : 11 - Button 1 (Ligado) - S Verde : True - S Am : False - S Vermelho : False
  Estado : 12 - Button 1 (Ligado) - S Verde : True - S Am : False - S Vermelho : False
  Estado : 13 - Button 1 (Ligado) - S Verde : True - S Am : False - S Vermelho : False
  Estado : 14 - Button 1 (Ligado) - S Verde : True - S Am : False - S Vermelho : False
  Estado : 15 - Button 1 (Ligado) - S Verde : True - S Am : False - S Vermelho : False
  Estado : 16 - Button 1 (Ligado) - S Verde : True - S Am : False - S Vermelho : False
  Estado : 17 - Button 1 (Ligado) - S Verde : True - S Am : False - S Vermelho : False
  Estado : 18 - Button 1 (Ligado) - S Verde : True - S Am : False - S Vermelho : False
  Estado : 19 - Button 0 (Ligado) - S Verde : True - S Am : False - S Vermelho : False
  Estado : 20 - Button 0 (Desligado) - S Verde : True - S Am : False - S Vermelho : False
  Estado : 21 - Button 0 (Desligado) - S Verde : True - S Am : False - S Vermelho : False
  Estado : 22 - Button 0 (Desligado) - S Verde : False - S Am : True - S Vermelho : False
```

MicroPyhon: Leitura Entrada Analógica



- **ADC – Analogic Digital Converter**

- **Conversor analógico digital 0 a 3.3 V**
- **12 Bits – 2^{12} – 4096 pontos**
- **Corresponde a 3.3 V / 4096 = 0,000805 V**

Programa para leitura entrada analógica:

```
from machine import Pin, ADC

pot = ADC(Pin(33))              # criação do objeto
pot.atten (ADC.ATTN_11DB)       # full range 3.3V

xx = pot.read()
print ('Valor ADC : ', xx   )
```

Exercício 6:

Ciclo com 100 leituras da entrada analógica, visualização do valor numérico e do valor convertido para tensão.

MicroPyhon: Conversão Tensão de entrada (ddp) em ºC

$$T = a_0 + a_1 v + a_2 v^2 \ldots + a_9 v^9 = \sum_{i=0}^{9} a_i v^i$$

Sensores para a medição de temperatura:

- Termopares – podem utilizar-se diferentes tipos de termopares, em função dos materiais utilizados (por exemplo Tipo K, Tipo J, etc.).
- Sondas de temperaturas resistivas (Ex. PT100)
- PTC e NTC (Sensores de temperatura baseados em materiais semicondutores)

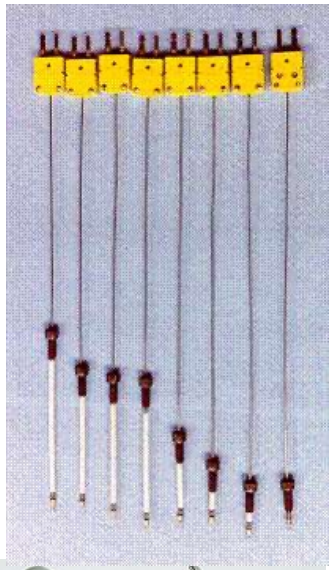


Tabela de tensões vs temperatura para um termopar tipo J, expressa em mV. A junção de referência está a 0ºC

ºC	0	1	2	3	4	5	6	7	8	9	10
0	0.000	0.050	0.101	0.151	0.202	0.253	0.303	0.354	0.405	0.456	0.507
10	0.507	0.558	0.609	0.660	0.771	0.762	0.813	0.865	0.916	0.967	1.019
20	1.019	1.070	1.122	1.174	1.225	1.277	1.329	1.381	1.432	1.484	1.536
30	1.536	1.588	1.640	1.693	1.745	1.797	1.849	1.901	1.954	2.006	2.058
40	2.058	2.111	2.163	2.216	2.268	2.321	2.374	4.426	2.479	2.532	2.585
50	2.585	2.638	2.691	2.743	2.796	2.849	2.902	2.956	3.009	3.062	3.115
60	3.115	3.168	3.221	3.275	3.328	3.381	3.435	3.488	3.542	3.595	3.649
70	3.649	3.702	3.756	3.809	3.863	3.917	3.971	4.024	4.078	4.132	4.186
80	4.186	4.239	4.293	4.347	4.401	4.455	4.509	4.563	4.617	4.671	4.725
90	4.725	4.780	4.834	4.888	4.942	4.996	5.050	5.105	5.159	5.213	5.268
100	5.268	5.322	5.376	5.431	5.485	5.540	5.594	5.649	5.703	5.758	5.812

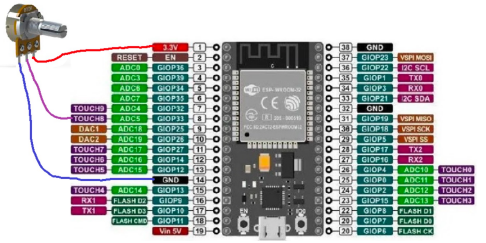
Coeficientes para diversos tipos de termopares

Exac.	*Tipo E* *– 100/1000°C* *± 0,5°C*	*Tipo J* *0/760°C* *± 0,1°C*	*Tipo K* *0/1370°C* *± 0,7°C*	*Tipo R* *0/1000°C* *± 0,5°C*	*Tipo S* *0/1750°C* *± 1°C*	*Tipo T* *– 160/400°C* *± 0,5°C*
a_0	0,1049673	– 0,0488683	0,2265846	0,2636329	0,9277632	0,1008609
a_1	17189,453	19873,145	24152,109	179075,491	169526,51	25727,944
a_2	282639,08	– 218614,54	67233,425	– 48840341,37	– 31568364	– 767345,83
a_3	12695339,5	11569199,8	2210340,7	1,90002E + 10	8990730663	78025596
a_4	– 448703085	– 264917531	– 860963915	– 4,8270E + 12	– 1,6356E + 12	– 9247486589
a_5	1,1086E + 10	2018441314	4,83506E + 10	7,62091E + 14	1,88027E + 14	6,97688E + 11
a_6	– 1,76807E + 11		– 1,18452E + 12	– 7,20026E + 16	– 1,3724E + 16	– 2,6619E + 13
a_7	1,71842E + 12		1,38690E + 13	3,71496E + 18	6,1750E + 17	3,9408E + 14
a_8	– 9,19278E + 12		– 6,33708E + 13	– 8,03104E + 19	– 1,56105E + 19	
a_9	2,06132E + 13				1,69535E + 20	

MicroPyhon: Conversão Tensão de entrada (ddp) em ºC

Conversão mV -> ºC (utilização de um termopar tipo J)

- O sinal deve ser amplificado de 15mV (para temperaturas entre os 0 e os 300ºC) -> 3,3V (tensão máxima da entrada do ESP32, com uma resolução de 12 bits, ou seja de 0 a 3,3V, corresponde uma variação entre os 0 e 4095)
- Utilizar um ganho de 200
- Na simulação não teremos este valor, simular com : AnalogIn [0..4095] = (In*4095)/3300 ;
- Converter em mV(0..3300mV] = (3300*AnalogIn)/4095;
- Converter no valor original sem amplificação: mV(2) = mV/200 ;



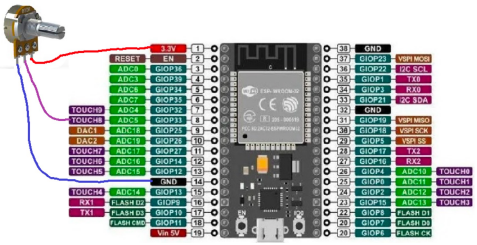
- Utilizar este valor no polinómio de conversão

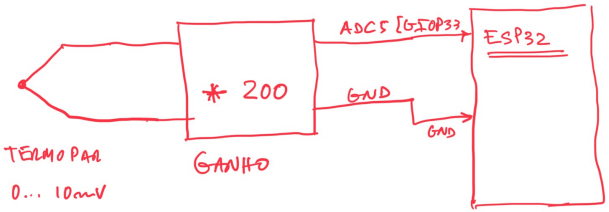
$$T = a_0 + a_1 v + a_2 v^2 \ldots + a_9 v^9 = \sum_{i=0}^{9} a_i v^i$$

Exercício 7:

Implementar o polinómio de conversão mV em ºC

MicroPyhon: Conversão Tensão de entrada (ddp) em ºC





$$T = a_0 + a_1 v + a_2 v^2 \ldots + a_9 v^9 = \sum_{i=0}^{9} a_i v^i$$

- O valor do termopar de 0..10mV, é amplificado 200*
- A ddp (TENSÃO) 0..3.3V DEVE SER DIVIDIDA POR 200, PARA PODERMOS APLICAR O POLINÓMIO DE CONVERSÃO

V → ºC

$T\cdot[^oC] = a_0 + a_1 V + a_2 V^2 + a_3 V^3 \ldots$

TERMOPAR TIPO K

$a_0 \simeq 0,2265846$

$a_1 = 24152,109$

$a_2 = 67233,425$

$a_3 = 2210340,7$

$a_4 = -86096395$

VALORES DA TENSÃO em V

PRG DE CONVERSÃO

1. LEITURA DA TENSÃO [V] → [XX]
2. DIVISÃO PELO GANHO [/200] [V= XX/200]
3. APLICAÇÃO DO POLINOMIO DE CONVERSÃO

```
C = a0 + a1 * V + a2 * (V**2) +
    a3 * (V**3) + a4 * (V**4) + ...
```

Exercício 7:

Implementar o polinómio de conversão mV em ºC

Python – ESP32

Notas Várias

IEA 2021-2022

A Borges

Sistemas de numeração

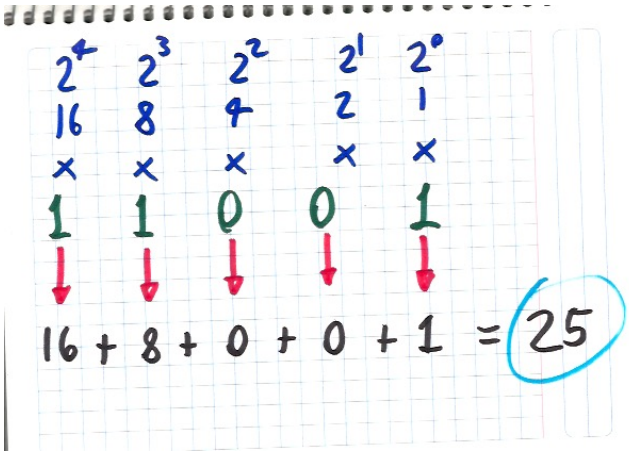
- **Sistemas de numeração:**
  - Decimal (Base 10: 0,1,2,3,4,5,6,7,8,9);
    $1998 = 1*10^3+9*10^2+9*10^1+8*10^0$
  - Binário (Base 2: 0,1);
  - Octal (Base 8: 0,1,2,3,4,5,6,7);
  - Hexadecimal (Base 16: 0,..9,A,B,C,D,E,F)

- Bit – unidade mínima de informação dos sistemas digitais;
  - **B**inary Dig**it** (BIT) (0 1) (2^1)
- Byte - 8 bits -> ($256 = 2^8$)

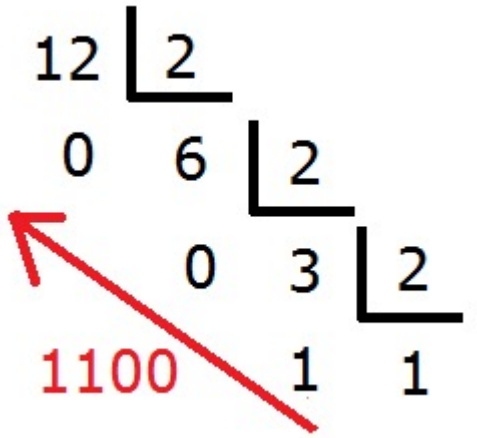
- 1 Byte = 8 bits
- 1 Kbyte = 1024 bytes
- 1 Mbyte = 1024 Kbytes
- 1 Gbyte = 1024 Mbytes
- 1 Tbyte = 1024 Gbytes

Sistemas de numeração

- **Conversão de binária/decimal:**



Conversão Binária->Decimal



Conversão Decimal->Binária

Bibliografia

Learn to Program with Python 3 - A Step-by-Step Guide to Programming -Second Edition

https://www.zerynth.com/zsdk/ [IDE-Zerinth]

https://www.open-electronics.org/python-on-esp32-easy-for-beginners-powerful-for-professionals/

https://thonny.org

https://www.w3schools.com/python/

https://docs.micropython.org

https://wokwi.com [Simulação de circuitos]